L13 - Inheritance

February 27, 2020

1 Inheritance

Inheritance is one of the major concepts in object-oriented programming, so it is important to understand. It will allow us to improve our code's structure.

1.1 Inheritance

1.1.1 The `network-v1` example

This example is a small social network. It will support news feed of text posts and photo posts. It allows us to search, display, and remove posts. `network-v1` does not use inheritance and indicates why we might want some.

1.1.2 `MessagePost.java`

[1]:
```java
import java.util.ArrayList;

/**
 * This class stores information about a post in a social network.
 * The main part of the post consists of a (possibly multi-line)
 * text message. Other data, such as author and time, are also stored.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.1
 */
public class MessagePost
{
    private String username;  // username of the post's author
    private String message;   // an arbitrarily long, multi-line message
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    /**
     * Constructor for objects of class MessagePost.
     *
     * @param author    The username of the author of this post.
     * @param text      The text of this post.
     */
    public MessagePost(String author, String text)
    {
        username = author;
        message = text;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
    }

    /**
     * Record one more 'Like' indication from a user.
     */
    public void like()
    {
        likes++;
    }

    /**
     * Record that a user has withdrawn his/her 'Like' vote.
     */
    public void unlike()
    {
        if (likes > 0) {
            likes--;
        }
    }

    /**
     * Add a comment to this post.
     *
     * @param text  The new comment to add.
     */
    public void addComment(String text)
    {
        comments.add(text);
    }

    /**
     * Return the text of this post.
     *
     * @return The post's text.
     */
    public String getText()
    {
        return message;
    }
    /**
     * Return the time of creation of this post.
     *
     * @return The post's creation time, as a system time value.
     */
    public long getTimeStamp()
    {
        return timestamp;
    }

    /**
     * Display the details of this post.
     *
     * (Currently: Print to the text terminal. This is simulating display
     * in a web browser for now.)
     */
    public void display()
    {
        System.out.println(username);
        System.out.println(message);
        System.out.print(timeString(timestamp));

        if(likes > 0) {
            System.out.println("  -  " + likes + " people like this.");
        }
        else {
            System.out.println();
        }

        if(comments.isEmpty()) {
            System.out.println("   No comments.");
        }
        else {
            System.out.println("   " + comments.size() + " comment(s). Click␣
↪here to view.");
        }
    }

    /**
     * Create a string describing a time point in the past in terms
     * relative to current time, such as "30 seconds ago" or "7 minutes ago".
     * Currently, only seconds and minutes are used for the string.
     *
     * @param time  The time value to convert (in system milliseconds)
     * @return      A relative time string for the given time
     */
    private String timeString(long time)
    {
        long current = System.currentTimeMillis();
        long pastMillis = current - time;      // time passed in milliseconds
        long seconds = pastMillis/1000;
        long minutes = seconds/60;
        if(minutes > 0) {
            return minutes + " minutes ago";
        }
        else {
            return seconds + " seconds ago";
        }
    }
}
```

1.1.3 PhotoPost.java

[2]:
```
import java.util.ArrayList;

/**
 * This class stores information about a post in a social network.
 * The main part of the post consists of a photo and a caption.
 * Other data, such as author and time, are also stored.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.1
 */
public class PhotoPost
{
    private String username;  // username of the post's author
    private String filename;  // the name of the image file
    private String caption;   // a one line image caption
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    /**
     * Constructor for objects of class PhotoPost.
     *
     * @param author    The username of the author of this post.
     * @param filename  The filename of the image in this post.
     * @param caption   A caption for the image.
     */
    public PhotoPost(String author, String filename, String caption)
    {
        username = author;
        this.filename = filename;
```

```java
        this.caption = caption;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
    }

    /**
     * Record one more 'Like' indication from a user.
     */
    public void like()
    {
        likes++;
    }

    /**
     * Record that a user has withdrawn his/her 'Like' vote.
     */
    public void unlike()
    {
        if (likes > 0) {
            likes--;
        }
    }

    /**
     * Add a comment to this post.
     *
     * @param text  The new comment to add.
     */
    public void addComment(String text)
    {
        comments.add(text);
    }

    /**
     * Return the file name of the image in this post.
     *
     * @return The post's image file name.
     */
    public String getImageFile()
    {
        return filename;
    }

    /**
     * Return the caption of the image of this post.
     *
     * @return The image's caption.
     */
    public String getCaption()
    {
        return caption;
    }

    /**
     * Return the time of creation of this post.
     *
     * @return The post's creation time, as a system time value.
     */
    public long getTimeStamp()
    {
        return timestamp;
    }

    /**
     * Display the details of this post.
     *
     * (Currently: Print to the text terminal. This is simulating display
     * in a web browser for now.)
     */
    public void display()
    {
        System.out.println(username);
        System.out.println("  [" + filename + "]");
        System.out.println("  " + caption);
        System.out.print(timeString(timestamp));

        if(likes > 0) {
            System.out.println("  -  " + likes + " people like this.");
        }
        else {
            System.out.println();
        }

        if(comments.isEmpty()) {
            System.out.println("   No comments.");
        }
        else {
            System.out.println("   " + comments.size() + " comment(s). Click␣
↪here to view.");
        }
    }

    /**
```

```
     * Create a string describing a time point in the past in terms
     * relative to current time, such as "30 seconds ago" or "7 minutes ago".
     * Currently, only seconds and minutes are used for the string.
     *
     * @param time  The time value to convert (in system milliseconds)
     * @return      A relative time string for the given time
     */

    private String timeString(long time)
    {
        long current = System.currentTimeMillis();
        long pastMillis = current - time;      // time passed in milliseconds
        long seconds = pastMillis/1000;
        long minutes = seconds/60;
        if(minutes > 0) {
            return minutes + " minutes ago";
        }
        else {
            return seconds + " seconds ago";
        }
    }
}
```

1.1.4 NewsFeed.java

[3]:
```
import java.util.ArrayList;

/**
 * The NewsFeed class stores news posts for the news feed in a social network
 * application.
 *
 * Display of the posts is currently simulated by printing the details to the
 * terminal. (Later, this should display in a browser.)
 *
 * This version does not save the data to disk, and it does not provide any
 * search or ordering functions.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.1
 */
public class NewsFeed
{
    private ArrayList<MessagePost> messages;
    private ArrayList<PhotoPost> photos;

    /**
     * Construct an empty news feed.
     */
    public NewsFeed()
    {
        messages = new ArrayList<>();
        photos = new ArrayList<>();
    }

    /**
     * Add a text post to the news feed.
     *
     * @param text  The text post to be added.
     */
    public void addMessagePost(MessagePost message)
    {
        messages.add(message);
    }

    /**
     * Add a photo post to the news feed.
     *
     * @param photo  The photo post to be added.
     */
    public void addPhotoPost(PhotoPost photo)
    {
        photos.add(photo);
    }

    /**
     * Show the news feed. Currently: print the news feed details to the
     * terminal. (To do: replace this later with display in web browser.)
     */
    public void show()
    {
        // display all text posts
        for(MessagePost message : messages) {
            message.display();
            System.out.println();   // empty line between posts
        }

        // display all photos
        for(PhotoPost photo : photos) {
            photo.display();
            System.out.println();   // empty line between posts
        }
    }
}
```

1.1.5 Demo

[9]:
```java
// create a new news feed
NewsFeed nf = new NewsFeed();

PhotoPost pp1 = new PhotoPost("Dave", "birb.jpg", "A birb");

nf.addPhotoPost(pp1);
pp1.addComment("birb");
pp1.like();
pp1.like();
pp1.like(); // everyone loves birbs

MessagePost mp1 = new MessagePost("Bill", "Nice birb");

nf.addMessagePost(mp1);
mp1.like();
mp1.like(); // nice wholesome post

nf.show();
```

```
Bill
Nice birb
0 seconds ago  -  2 people like this.
   No comments.

Dave
  [birb.jpg]
  A birb
0 seconds ago  -  3 people like this.
   1 comment(s). Click here to view.
```

We should note that a bunch of the fields and methods of both `MessagePost` and `PhotoPost` are the same: `username`, `likes`, `comments`, and `timestamp` are common fields, `like()`, `unlike()`, `addComment()`, `getTimeStamp()`, `display()` and the `timeString` helper are common methods.

The overall network model has an `ArrayList` of `MessagePosts` and `PhotoPosts`; `NewsFeed` includes both `MessagePost` and `PhotoPost`.

1.2 Using inheritance

Because of all the duplicated functions, we could make a new *superclass* `Post` that defines the common attributes like Likes and Comments, then have *subclasses* `PhotoPost` and `MessagePost` that would **inherit** these methods from the superclass.

This has a number of advantages - we duplicate less code, so we can fix bugs in one place instead of in each subclass. Extending the superclass benefits all its subclasses. Each subclass only needs to define its unique functionality, which means they are more readable.

1.3 Inheritance Hierarchy

These benefits can extend several levels deep. For example, "Polly" might inherit from "Parrot" which might inherit from "Bird" which might inherit from "Animal".

Note that the terms "superclass" and "subclass" are relative; "Bird" would be a subclass of "Animal" but a superclass of "Parrot".

Hierarchies can be arbitrarily deep. You can add another level of class to control what methods a subclass can inherit, for example, add a `TalkingBird` class if you don't want `Sparrow` inheriting your `getWordsKnown()` method.

1.3.1 `extends`

Java uses the `extends` keyword to describe a class inheriting from a superclass:

[39]:
```
public class Animal
{

    public Integer number_of_feet;

    public Animal()
    {
        number_of_feet = 0;
    }

    public Integer getFeet()
    {
        return number_of_feet;
    }
}

public class Cat extends Animal
{
    public Cat()
    {
        super();
        this.number_of_feet = 4;
    }
}

public class Bird extends Animal
{
    public Bird()
    {
        super();
        this.number_of_feet = 2;
    }
}
```

[40]:
```
Bird tweety = new Bird();

Cat sylvester = new Cat();

sylvester.getFeet();
```

[40]:
```
4
```

[41]:
```
tweety.getFeet();
```

[41]:
```
2
```

1.3.2 super()

Calls the constructor of a superclass. Subclass constructors *must always* contain a `super()` call as the *first* statement in the subclass constructor. If you don't write one, the compiler will make one, but will only succeed if the superclass has a default constructor (without parameters).

1.4 network-v2

1.4.1 Post

[25]:
```
import java.util.ArrayList;

/**
 * This class stores information about a news feed post in a
 * social network. Posts can be stored and displayed. This class
 * serves as a superclass for more specific post types.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.2
 */
public class Postv2
{
    private String username;  // username of the post's author
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    /**
     * Constructor for objects of class Post.
     *
     * @param author    The username of the author of this post.
     */
    public Postv2(String author)
    {
        username = author;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
    }

    /**
     * Record one more 'Like' indication from a user.
     */
    public void like()
    {
        likes++;
    }

    /**
     * Record that a user has withdrawn his/her 'Like' vote.
     */
    public void unlike()
    {
        if (likes > 0) {
            likes--;
        }
    }

    /**
     * Add a comment to this post.
     *
     * @param text  The new comment to add.
     */
    public void addComment(String text)
    {
        comments.add(text);
    }

    /**
     * Return the time of creation of this post.
     *
     * @return The post's creation time, as a system time value.
     */
    public long getTimeStamp()
    {
        return timestamp;
    }

    /**
     * Display the details of this post.
     *
     * (Currently: Print to the text terminal. This is simulating display
     * in a web browser for now.)
     */
    public void display()
    {
        System.out.println(username);
        System.out.print(timeString(timestamp));

        if(likes > 0) {
            System.out.println("  -  " + likes + " people like this.");
        }
        else {
            System.out.println();
        }

        if(comments.isEmpty()) {
            System.out.println("   No comments.");
        }
        else {
            System.out.println("   " + comments.size() + " comment(s). Click␣
↪here to view.");
        }
    }

    /**
     * Create a string describing a time point in the past in terms
     * relative to current time, such as "30 seconds ago" or "7 minutes ago".
     * Currently, only seconds and minutes are used for the string.
     *
     * @param time  The time value to convert (in system milliseconds)
     * @return      A relative time string for the given time
     */

    private String timeString(long time)
    {
        long current = System.currentTimeMillis();
        long pastMillis = current - time;      // time passed in milliseconds
        long seconds = pastMillis/1000;
        long minutes = seconds/60;
        if(minutes > 0) {
            return minutes + " minutes ago";
        }
        else {
            return seconds + " seconds ago";
        }
    }
}
```

1.4.2 MessagePost

[28]:
```java
import java.util.ArrayList;

/**
 * This class stores information about a post in a social network news feed.
 * The main part of the post consists of a (possibly multi-line)
 * text message. Other data, such as author and time, are also stored.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.2
 */
public class MessagePostv2 extends Postv2
{
    private String message;  // an arbitrarily long, multi-line message

    /**
     * Constructor for objects of class MessagePost.
     *
     * @param author    The username of the author of this post.
     * @param text      The text of this post.
     */
    public MessagePostv2(String author, String text)
    {
        super(author);
        message = text;
    }

    /**
     * Return the text of this post.
     *
     * @return The post's message text.
     */
    public String getText()
    {
        return message;
    }
}
```

1.4.3 PhotoPost

[29]:
```java
import java.util.ArrayList;

/**
 * This class stores information about a post in a social network news feed.
 * The main part of the post consists of a photo and a caption.
 * Other data, such as author and time, are also stored.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.2
 */
public class PhotoPostv2 extends Postv2
{
    private String filename;  // the name of the image file
    private String caption;   // a one line image caption

    /**
     * Constructor for objects of class PhotoPost.
     *
     * @param author    The username of the author of this post.
     * @param filename  The filename of the image in this post.
     * @param caption   A caption for the image.
     */
    public PhotoPostv2(String author, String filename, String caption)
    {
        super(author);
        this.filename = filename;
        this.caption = caption;
    }

    /**
     * Return the file name of the image in this post.
     *
     * @return The post's image file name.
     */
    public String getImageFile()
    {
        return filename;
    }

    /**
     * Return the caption of the image of this post.
     *
     * @return The image's caption.
     */
    public String getCaption()
    {
        return caption;
    }
}
```

1.4.4 NewsFeed

[32]:
```java
import java.util.ArrayList;

/**
 * The NewsFeed class stores news posts for the news feed in a
 * social network application.
 *
 * Display of the posts is currently simulated by printing the
 * details to the terminal. (Later, this should display in a browser.)
 *
 * This version does not save the data to disk, and it does not
 * provide any search or ordering functions.
 *
 * @author Michael Kölling and David J. Barnes
 * @version 0.2
 */
public class NewsFeedv2
{
    private ArrayList<Postv2> posts;

    /**
     * Construct an empty news feed.
     */
    public NewsFeedv2()
    {
        posts = new ArrayList<>();
    }

    /**
     * Add a post to the news feed.
     *
     * @param post  The post to be added.
     */
    public void addPost(Postv2 post)
    {
        posts.add(post);
    }

    /**
     * Show the news feed. Currently: print the news feed details
     * to the terminal. (To do: replace this later with display
     * in web browser.)
     */
    public void show()
    {
        // display all posts
        for(Postv2 post : posts) {
            post.display();
            System.out.println();   // empty line between posts
        }
    }
}
```

1.4.5 Demo

[42]:
```
// create a new news feed
NewsFeedv2 nf2 = new NewsFeedv2();

PhotoPostv2 pp2 = new PhotoPostv2("Dave", "birb.jpg", "A birb");

nf2.addPost(pp2);
pp2.addComment("birb");
pp2.like();
pp2.like();
pp2.like(); // everyone loves birbs

MessagePostv2 mp2 = new MessagePostv2("Bill", "Nice birb");

nf2.addPost(mp2);
mp2.like();
mp2.like(); // nice wholesome post

nf2.show();
```

```
Dave
0 seconds ago  -  3 people like this.
   1 comment(s). Click here to view.

Bill
0 seconds ago  -  2 people like this.
   No comments.
```

Note that the new `NewsFeed` is less complex because we don't need to handle two types anymore - instead, we can treat both `MessagePostv2` and `PhotoPostv2` as `Postv2` and put them in a single list, removing on efo the loops from the `show()` method.

1.5 Subtypes and subclasses

Subclasses define subtypes. If you have a superclass, you can always use an object of the subtype in place of an object of the superclass (the subclass always extends the superclass). You *cannot* go the other way.

[ ]:
```
// subtyping example
```

```
Vehicle v1 = new Vehicle()
Vehicle v2 = new Car(); // if Car inherits from vehicle, this is legit
Vehicle v3 = new Bicycle(); // also legit if Bicycle is a subclass of Vehicle
```

Subclass objects can be used as actual parameters for the superclass.

1.6 Polymorphic Variables

Object variables in Java can hold objects of more than one type. They can hold objects of the declared type, or of subtypes of the declared type.

1.6.1 Casting operator

We can assign a subtype to a supertype, but we cannot assign supertypes to subtypes.

```
Vehicle v;

Car c = new Car();
v = c; // works
c = v; // error

c = (Car) v; // use the casting operator to fix
```

You can only do this if `c` is actually a `Car`.

1.7 `Object`

All Java classes inherit from `Object`. `Object` is the universal class that makes up all classes.

[ ]:

```
```